
Dasher!!

:warning: Dasher is sunsetted :warning:

This project is no longer under active development. I neither use it, nor do I use Amazon Dash buttons in my home automation. I found their latency and general way of working to not be a great solution as a smart button for my household.

Nekmo/amazon-dash is a great alternative. It has support for all kinds of actions and is actively developed. It's way better than mine!

What it is

Dasher is a simple way to bridge your Amazon Dash buttons to HTTP services.

Do you have a Home Automation service set up like Home Assistant, openHab, or maybe a Smart-Things hub? Using Dasher, you can easily command them to do something whenever your Dash button is pressed.

This of course goes for anything you can reach via HTTP. That includes IFTTT by way of the Maker channel :metal:

How it works

It's pretty simple. Press a button and an HTTP request is made or local command is ran. That's it.

You configure your Dash button(s) via `config/config.json`. You add its network address and either a url, an http method, and optionally a content body and headers or a local command to execute.

When Dasher starts, it will listen for your button being pressed. Once it sees it, it will then make the HTTP request or run the command that you defined for it in your config.

Configuration

You define your buttons via the `config/config.json` file. It's a simple JSON file that holds an array of buttons.

Here's an example.

```

1 {"buttons":[
2   {
3     "name": "Notify",
4     "address": "43:02:dc:b2:ab:23",
5     "interface": "en0",
6     "timeout": "60000",
7     "protocol": "udp",
8     "url": "https://maker.ifttt.com/trigger/Notify/with/key/5212
        ssx2k23k2k",
9     "method": "POST",
10    "json": true,
11    "body": {"value1": "any value", "value2": "another value", "value3"
        : "wow, even more value"}
12  },
13  {
14    "name": "Party Time",
15    "address": "d8:02:dc:98:63:49",
16    "url": "http://192.168.1.55:8123/api/services/scene/turn_on",
17    "method": "POST",
18    "headers": {"authorization": "your_password"},
19    "json": true,
20    "body": {"entity_id": "scene.party_time"},
21    "formData": {
22      "var1": "val1",
23      "var2": " val2"
24    }
25  },
26  {
27    "name": "Start Cooking Playlist",
28    "address": "66:a0:dc:98:d2:63",
29    "url": "http://192.168.1.55:8181/playlists/cooking/play",
30    "method": "PUT"
31  },
32  {
33    "name": "Debug Dash Button",
34    "address": "41:02:dc:b2:ab:23",
35    "debug": true
36  },
37  {
38    "name": "Command Exec Button",
39    "address": "41:02:dc:b2:10:12",
40    "cmd": "/home/user/dash_button.sh"
41  }
42 ]}]

```

Buttons take up to 8 options.

- **name** - Optionally give the button action a name.
- **address** - The MAC address of the button.
- **interface** - Optionally listen for the button on a specific network interface. (**enX** on OS X and

ethX on Linux)

- `timeout` - Optionally set the time required between button press detections (if multiple presses are detected) in milliseconds. Default is 5000.
- `protocol` - Optionally set the protocol for your Dash button. Options are udp, arp, and all. Default listens to arp. The “newer” JK29LP button from ~Q2 2016+ tends to use udp.
- `url` - The URL that will be requested.
- `method` - The HTTP method of the request.
- `headers` - Optional headers to use in the request.
- `json` - Optionally declare the content body as being JSON in the request.
- `body` - Optionally provide a content-body that will be sent with the request.
- `formData` - Optionally add formData that will be sent with the request.
- `debug` - Used for testing button presses and will -not- perform a request.
- `cmd` - Used to run a local command rather than an HTTP request. Setting this will override the url parameter.

Setting and using these values should be enough to cover almost every kind of request you need to make.

You can find more examples in the example config.

Protips

Here are few protips about Dash buttons that will help you plan how to use them.

- Dash buttons take ~5 seconds to trigger your action.
- Use DHCP Reservation on your Dash button to lower the latency from ~5s to ~1s.
- Dash buttons are discrete buttons. There is no on or off. They just do a single command.
- Dash buttons can not be used for another ~10 seconds after they’ve been pressed.
- If your Dash button is using udp, specify it in the button config.
- Listening over wifi is unreliable. I highly recommend using ethernet, especially on Raspberry Pi

Dash buttons should be used to trigger specific things. I.E. a scene in your home automation, as a way to turn everything off in your house, or as a simple counter.

Setup

You’ll want to set up your Dash buttons as well as Dasher.

Dash button

Setting up your Dash button is as simple as following the instructions provided by Amazon **EXCEPT FOR THE LAST STEP**. Just follow the instructions to set it up in their mobile app. When you get to the step where it asks you to pick which product you want to map it to, just quit the setup process.

The button will be set up and available on your network.

Find Dash Button Once your Dash button is set up and on your network, you need to determine its MAC address. Run this:

```
1 script/find_button
```

Click your Dash button and the script will listen for your device. Dash buttons should appear as manufactured by 'Amazon Technologies Inc.'. Once you have its MAC address you will be able to configure it in Dasher by modifying `config/config.json` after installing Dasher.

Dasher app

Simply **install the dependencies** and **clone the repository**.

note: You might need to install `libpcap-dev` or `npm` on Linux first.

```
1 sudo apt-get install libpcap-dev
2 sudo apt-get install npm
```

note Raspberry Pi users may need to update node arm which will automatically remove nodejs-legacy. Credit @legotheboss

```
1 sudo apt-get install node
2
3 wget http://node-arm.herokuapp.com/node_latest_armhf.deb
4 sudo dpkg -i node_latest_armhf.deb
```

Clone and Set up Dasher

```
1 git clone https://github.com/maddox/dasher.git
2 cd dasher
3 npm install
```

Then create a `config.json` in `/config` to set up your Dash buttons. Use the example to help you. If you just want to test the button press, use the debug button example with the MAC address you found running `script/find_button`.

Running It

Listening for Dash buttons requires root. So you need to launch Dasher with sudo.

```
1 sudo npm run start
```

Auto Start on OS X

After setting it up with `script/bootstrap` just run `script/install` to load Dasher with `launchd`. Dasher will now start on boot.

You can uninstall it with `script/uninstall` and restart it with `script/restart`.

Raspberry Pi

Having problems running npm install?

Raspberry Pi users may need to update node arm which will automatically remove nodejs-legacy. One you are on node mainline, force the update to the latest version of node for arm. This may not be needed if you are running a first generation pi. If you are on a Pi and run into problems with npm install, try this. Credit @legotheboss

Replace nodejs-legacy with node and manually update to the latest version of node arm.

```
1 sudo apt-get install node
2 wget http://node-arm.herokuapp.com/node_latest_armhf.deb
3 sudo dpkg -i node_latest_armhf.deb
```

Quick Start Works as of (1/27/17)

Starting from a fresh Raspberry Pi Build?

```
1 sudo apt-get install libpcap-dev
2 sudo apt-get install npm
3
4 sudo apt-get install node
5 wget http://node-arm.herokuapp.com/node_latest_armhf.deb
6 sudo dpkg -i node_latest_armhf.deb
7
8 git clone https://github.com/maddox/dasher.git
9 cd dasher
10 sudo npm install
11
12 sudo ./script/find_button
13 # update /config/config.json with mac address of your button
14 sudo npm run start
```

Auto Starting

Advanced information on autostarting Dasher on your Raspberry Pi can be found [here](#).

Contributions

- fork
- create a feature branch
- open a Pull Request